CC3301 Programación de Software de Sistemas – Control 1
Semestre Otoño 2018 – Prof.: Luis Mateu

Pregunta 1

Parte a.- Programe la función *compr* con el siguiente encabezado:

```
typedef unsigned int uint;
uint compr(uint *a, int nbits);
```

Esta función comprime múltiples enteros sin signo almacenados en el arreglo *a* en un solo entero sin signo. Para ello se retorna la concatenación de todos los elementos en *a* truncados a *nbits*. La cantidad de elementos almacenados en el arreglo *a* es el número de enteros de *nbits* que caben en un entero sin signo, es decir el máximo *k* que cumple con *k*nbits<=sizeof(uint)*8*, en donde *sizeof(uint)*8* es el tamaño de un entero sin signo (32 bits en la mayoría de las plataformas).

Ejemplo de uso:

```
uint a[]= { 0b 100 110 101 011 000, 0b 000 101 101 011, 0b 100 001 010 000 };
uint r= compr(a, 9); // r es 0b 101 011 000  101 101 011  001 010 000
                     //         <- a[0]  ->  <- a[1]  ->  <- a[2]  ->
```

Observe que al truncar *a[0]* a 9 bits se perdieron los bits más significativos de *a[0]*. Con el fin de facilitar la comprensión se empleó la notación *0b...* para expresar números en base 2, pero no es parte del lenguaje C. Considere que el parámetro *nbits* puede variar entre 1 y *sizeof(uint)*8-1*.

Restricción: Ud. no puede usar los operadores de multiplicación, división o módulo (* / %). Use eficientemente los operadores de bits.

Parte b.- Programe la siguiente función:

```
void elimEspacios(char *s);
```

Esta función reemplaza múltiples espacios contiguos en el string *s* por un solo espacio. Ejemplo de uso:

```
char s[]= "hola     que     tal";
elimEspacios(s); //s es "hola que tal"
```

El string *s* no tiene espacios en blanco al final o al principio.

Restricción: Ud. no puede usar el operador de subindicación [ ] ni su equivalente **(p+i)*. Para recorrer el string use el operador ++. Use múltiples punteros para direccionar distintas partes del string. No olvide terminar *s*.

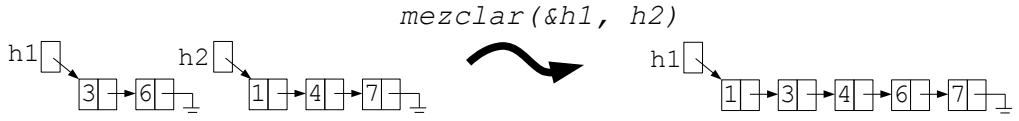
Pregunta 2

Programe la función *mezclar* que une 2 listas simplemente enlazadas ordenadas ascendentemente. La lista resultante también debe quedar ordenada ascendentemente. Su declaración es:

```
typedef struct nodo {
  int x;
  struct nodo *prox;
} Nodo;

void mezclar(Nodo **ph1, Nodo *h2);
```

La siguiente figura muestra el resultado de invocar *mezclar(&h1, h2)*. Los punteros *h1* y *h2* son de tipo *Nodo**.



Restricciones: Ud. no puede usar ninguna forma de iteración (while, for, etc.). Ud. debe usar recursividad (¡la versión no recursiva es complicada!). Ud. no puede usar *malloc*. Reutilice los nodos de los argumentos.